Report of Comp1

I. Regression analysis:

Introduction: Data given to you in *Comp1_IE529* contains two vectors, 'putt_m' and 'lift_kg', where 'putt_m(i)' corresponds to the longest shot-put by athlete i in meters., and 'lift_kg(i)' corresponds to the maximum weight lifted by athlete i in kilograms. Your assignment is to use regression methods to determine a model that describes the relationship between the two variables. That is, suppose x_1 = 'lift' and x_2 = 'putt'; you should find a mathematical model relating x_1 and x_2. One example of such a model could be

$$x_2 = 10 + 2x_1 + x_1^2 - 0.1x_1^3.$$

The relationship may be linear/affine, polynomial or logistic.

1. **Write a simple program** to compute a least squares solution for the case of linear or polynomial regression, and determine the lowest order model that fits the data reasonably well (order 1 is linear, and higher orders are polynomial).
2. Consider using an existing logistic regression function in Matlab or Python (or another alternative if you so choose, but please let us know) to determine if a logistic model will fit the data better, or not. Discuss.
3. State which of your candidate models best describes the data, taking into account that *simplicity is preferred.* **Provide plots** of (1) a linear fit, (2) your best polynomial fit (i.e., second order or higher) and (3) if possible one logistic fit (and otherwise discuss results here). Compute the sum-of-square of residuals of these models, i.e., give the cost $\|\epsilon_i\|_2^2$, for each of the models plotted.
4. In your report include (a) clearly labeled plots with associated explicit models and costs, (b) a brief discussion explaining your model choice, and (c) pseudocode and/or a description of your code/function calls.

1. Pseudocode of linear regression-

1. Convert csv into numpy array X,y
2. Calculate the least square estimator 'm' of β
3. Use m, X,y to compute 'b', which is an unbiased estimator of α
4. Calculate the forcasted data 'y_pred' derived from the model
5. Calculate the sum of squares of the residuals 'SSR'

Pseudocode of polynomial regression-

1. Convert csv into numpy array X,y
2. Add a zero vector and new vectors in X accroding to the degrees of polynomial regression model
3. Compute β by the fomula $\boldsymbol{\beta} = (X^T X)^{-1} X^T \mathbf{y}$
4. Calculate the forcasted data 'y_pred' by X times β
5. Calculate the sum of square of the residual 'SSR'

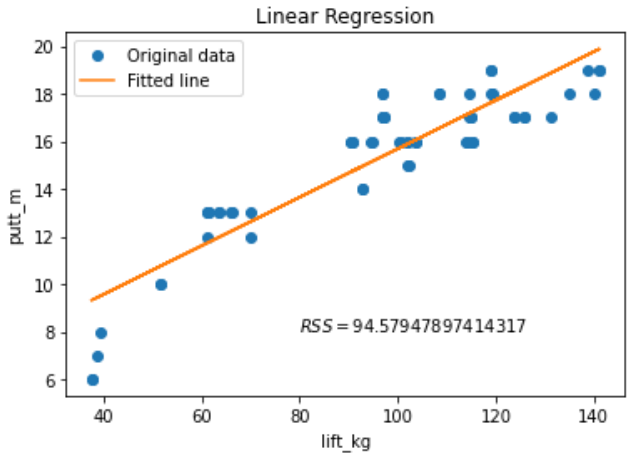
From the SSR results of linear regression and polynomila regressions shown in graphs of question3, third-order polynomial regression model fits the data reasonably well with a relatively low order.

2. Logistic regression is used in classification problems, like binary classfication problem and muticlass problem. Our data here only contains continuous values but not discrete values, so a logistic regression model cannot directly fit the data.

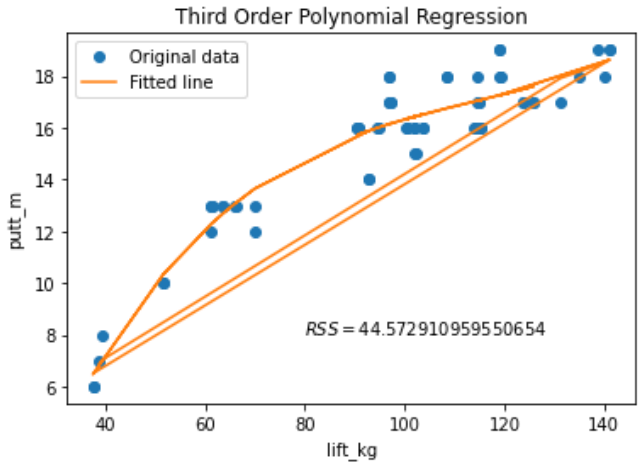
To try using logistic regression, I rounded the value of 'putt_m', and converted the data into muticlass data. I use the logistic function with the solver"newton-cg",which only support l2 regularization or no regularization, and are found to converge faster for some high-dimensional data. I had also tried other solvers such as "lbfgs","sag", which are also used for muticlass problem, but they cannot converge or have a large RSS.

3. The third order polynomial regression best describes the data with simplicity.
RSS: 44.573 (the third order polynomial regression) > 58.0 (logistic regression) > 94.579 (linear regression)

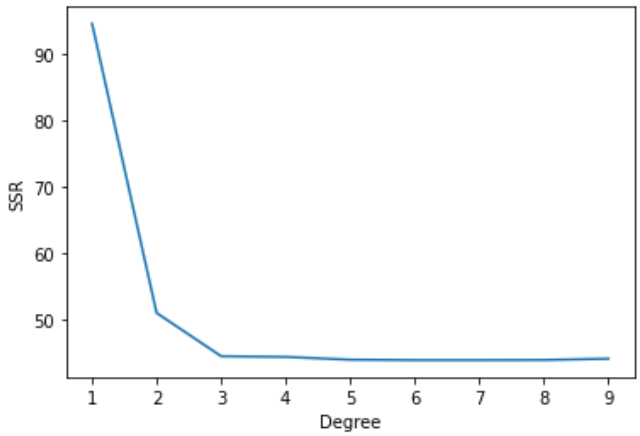
(1) Linear fit



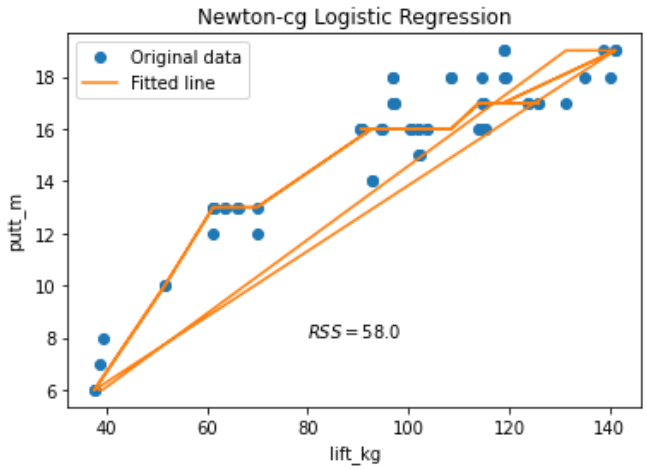
(2) Best polynomial fit



SSR changes with degrees of polynomial



(3) Logistic fit



II. Principal Component Analysis: real data

Introduction: Data for this portion of the assignment (PCA_comp1.mat and PCA_comp1.csv) consists of 4 column vectors of data, each with 150 entries (NOTE: **each row is one sample** with 4 entries; the number of rows is the number of samples). Each column represents a measurement, in centimeters, of one specific feature, taken from a sample of 150 flowers. The four features are sepal length, sepal width, petal length and petal width. We would like to distill this data down to a lower dimension (2 or 3), **and** try to determine how many species might be represented by the data.

1. For the given data, de-mean each entry, using **column** sample means. Perform a PCA on the de-meaned data. Clearly explain how you performed the PCA (submit pseudocode for your own PCA code, or reference and describe any function you called, i.e, from what library, how it works, what it takes as inputs and produces as outputs).
2. State what portion of the variance in the data is contained in each of the 4 principal components. From these values, state how many *true* components are needed to represent the data.
3. On a 2D graph with the horizontal axis given by the first PC, and the vertical plot given by the second PC, plot the 2D representation of all the data points (i.e., find the 2D projection of each row). Discuss: (a.) are there any visually apparent clusters, if so, how many, and (b.) from this do you think you can conjecture anything about the number of species represented by the data?
4. Repeat parts 1- 3 for *standardized* data: in addition to de-meaning the entries by column means, you should also scale by the inverse of the sample standard deviation, i.e., for each element x_{ij} in the original matrix X, normalize the element to $\tilde{x}_{ij}$ where

$$\tilde{x}_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j}.$$

   Note whether or not this scaling changes the outcome of your analysis.
5. In your report include (a) a matrix with the 4 components for the data and their associated variances, and (b) plots for parts 3 and 4 (i.e., for both the de-meaned and the standardized data sets).

1.

Pseudocode:

1. subtract off the mean for each dimension
2. divided by sqrt(N-1) and construct the matrix Y
3. do SVD in matrix Y
4. calculate the variances

Functions called in the source code:

Size: `[sz1,…,szN] = size(A)` returns the lengths of the queried dimensions of matrix `A` separately.

(https://www.mathworks.com/help/matlab/ref/size.html)

Mean: `M = mean(A,dim)` returns the mean along dimension `dim`. For example, if `A` is a matrix, then `mean(A,2)` is a column vector containing the mean of each row.
(https://www.mathworks.com/help/matlab/ref/mean.html)

Repmat: B = repmat(A,n) returns an array containing n copies of A in the row and column dimensions. The size of B is size(A)*n when A is a matrix.
(https://www.mathworks.com/help/matlab/ref/repmat.html)

Sqrt: `B = sqrt(X)` returns the square root of each element of the array `X`. For the elements of `X` that are negative or complex, `sqrt(X)` produces complex results.
(https://www.mathworks.com/help/matlab/ref/sqrt.html)

Svd: `[U,S,V] = svd(A)` performs a singular value decomposition of matrix `A`, such that `A = U*S*V'`.
U – Left singular vectors; S – Singular values; V – Right singular vectors
(https://www.mathworks.com/help/matlab/ref/double.svd.html)

Diag: `D = diag(v)` returns a square diagonal matrix with the elements of vector `v` on the main diagonal.
(https://www.mathworks.com/help/matlab/ref/diag.html)

2. Portions of variances in 4 principal components-

Por

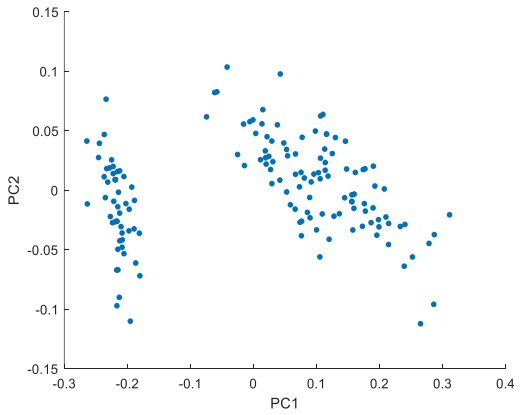
4x1 double

	1
1	0.9246
2	0.0530
3	0.0172
4	0.0052

PC1: 0.924616207174268
PC2: 0.0530155678505351
PC3: 0.0171851395250068
PC4: 0.00518308545018993

Typically, we choose PCs that can explain at least 99% variance, so we can choose 3 principal components in this case.

3.



(a) There are two visible clusters.
(b) I can conjecture that there might be totally two species represented by the data.

4.

4.1

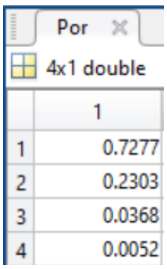
Pseudocode:

1. subtract off the mean for each dimension
2. scale by the inverse of the sample standard deviation
3. divided by sqrt(N-1) and construct the matrix Y
4. do SVD in matrix Y
5. calculate the variances

Functions called in the source code:

Std: `S = std(A)` returns the standard deviation of the elements of `A` along the first array dimension whose size does not equal 1.
(https://www.mathworks.com/help/matlab/ref/sqrt.html)

4.2



PC1: 0.727704520938013

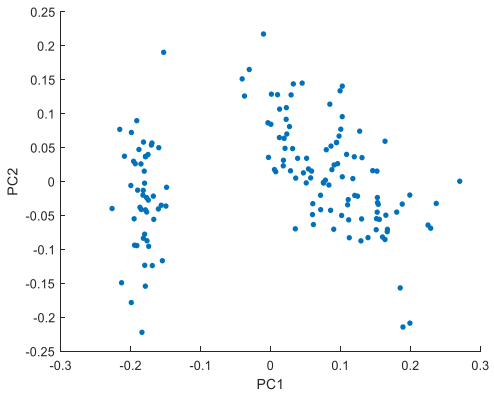
PC2: 0.230305232676807

PC3: 0.0368383195762739

PC4: 0.00515192680890631

Typically, we choose PCs that can explain at least 99% variance, so we can choose 3 principal components in this case.

4.3



We can notice from the two graphs that this scaling has changed the outcome of the analysis. The distribution range of each point in PC2 is expanded, which verifies that the portion of variance of PC2 has increased from 0.053 to 0.230.

(a) There are two visible clusters.
(b) I can conjecture that there might be totally two species represented by the data.

appendix

1. Regression analysis (Python)

```python
import numpy as np
from matplotlib import pyplot as plt
import pandas as pd

# load the dataset
url='https://drive.google.com/file/d/1C3Lwh_lQwpJQAkT-ls3IR87Fq3M_JOPI/view?usp=sharing'
url2='https://drive.google.com/uc?id=' + url.split('/')[-2]
df = pd.read_csv(url2,names=['lift_kg','putt_m']);

# linear regression
xy=df.to_numpy()
X=xy[:,0]
y=xy[:,1]
m=(np.sum((X-np.mean(X))*y))/(np.sum(X*X)-(len(X)*((np.mean(X))**2))) # Calculate the least square estimator ‘m’ of β
b=np.sum(y/len(X)) - m*np.mean(X) # Use m, X,y to compute ‘b’, which is an unbiased estimator of α
y_pred=m*X+b # Calculate the forcasted data ‘y_pred’ derived from the model
SSR=np.sum(np.square(y-y_pred)) # Calculate the sum of squares of the residuals ‘SSR’
p1=plt.plot(X,y,'o',label='Original data')
p1=plt.plot(X,m*X+b,label='Fitted line')
p1=plt.legend()
p1=plt.xlabel('lift_kg')
p1=plt.ylabel('putt_m')
p1=plt.title('Linear Regression')
p1=plt.text(80, 8, r'$RSS=94.579478974143 17$')
plt.show()

# polynomial regression(fit third-order model)
X=xy[:,0]
y=np.transpose(xy[:,1])
X = np.c_[np.ones(X.shape[0]), X,  X**2, X**3] # dd a zero vector and new vectors in X accroding to the degrees of polynomial regression model

# Compute beta
Xt = np.transpose(X)
XtX = np.dot(Xt,X)
Xty = np.dot(Xt,y)
beta = np.linalg.solve(XtX,Xty) # Compute β by the fomula
print(beta)

y_pred=np.dot(X,beta) # Calculate the forcasted data ’y_pred’ by X times β
SSR=np.dot((y-y_pred).T,(y-y_pred)) # Calculate the sum of square of the residual ‘SSR’

X=xy[:,0]
# plot of the best polynomial fit
p3=plt.plot(X,y,'o',label='Original data')
p3=plt.plot(X,y_pred,label='Fitted line')
p3=plt.xlabel('lift_kg')
```

```
p3=plt.ylabel('putt_m')
p3=plt.title('Third Order Polynomial Regression')
p3=plt.text(80, 8, r'$RSS=44.5729109595506548')
p3=plt.legend()
plt.show()

# find the best degree of polynomial
xy=df.to_numpy()
X=xy[:,0].reshape(-1,1)
y=xy[:,1]
# program to compute a least square solution
def get_les(reg,X_input):
    y_pred=reg.predict(X_input)
    LES=np.sum(np.square(y-y_pred))
    return LES

from sklearn.preprocessing import PolynomialFeatures

les=[]
min_les,min_deg=0,0
degrees=np.arange(1,10)
for deg in degrees:
    poly_features=PolynomialFeatures(degree=deg,include_bias=False)
    X_poly=poly_features.fit_transform(X)
    poly_reg=LinearRegression()
    poly_reg.fit(X_poly,y)

    poly_les=get_les(poly_reg,X_poly)
    les.append(poly_les)

fig=plt.figure()
p2=fig.add_subplot(111)
p2.plot(degrees,les)
p2.set_xlabel('Degree')
p2.set_ylabel('SSR')

# logistic regression
df['putt_m']=round(df['putt_m'])
x=df.drop('putt_m',axis=1)
y=df['putt_m']

from sklearn.linear_model import LogisticRegression

log_reg=LogisticRegression(solver='newton-
cg',multi_class='multinomial') # new 'newton-cg' solver for muticlass data
log_reg.fit(x,y)
y_pred=log_reg.predict(x)

p4=plt.plot(x,y,'o',label='Original data')
p4=plt.plot(x,y_pred,label='Fitted line')
p4=plt.legend()
p4=plt.xlabel('lift_kg')
p4=plt.ylabel('putt_m')
p4=plt.title('Newton-cg Logistic Regression')
p4=plt.text(80, 8, r'$RSS=58.0$')
plt.show()
```

2.Principal Component Analysis: real data (Matlab)

(1) for only de-meaned data

```
df=PCA_comp1;
```

```
[M,N]=size(df);

% subtract off the mean for each dimension
mn=mean(df,1);
dfdf=df-repmat(mn,M,1);

% construct the matrix Y
Y=df'/sqrt(M-1);

% SVD does it all
[u,s,PC]=svd(Y);

% calculate the variances
S=diag(s);
V=S.*S;

% project the original data
signals=PC'*df;

% obtain portions of variances
Por=V/sum(V);

% typically we choose PCs that can explain 99% varinace, so we can
% choose 3PC in this case

C = s(1:4,1:4)*PC(:,1:4)';
figure(2);
scatter(C(1,:),C(2,:),17,'filled')
xlabel('PC1'); ylabel('PC2');
```

(2) for de-meaned and standardized data

```
df=PCA_comp1;
% subtract off the mean for each dimension
mn=mean(df,1);
dfdf=df-repmat(mn,M,1);

% scale by the inverse of the sample standard deviation
dfdf=df./std(df);

[M,N]=size(df);

% construct the matrix Y
Y=df'/sqrt(M-1);

% SVD does it all
[u,s,PC]=svd(Y);

% calculate the variances
S=diag(s);
V=S.*S;

% project the original data
signals=PC'*df;

% obtain portions of variances
Por=V/sum(V);

```

```
C = s(1:4,1:4)*PC(:,1:4)';
figure(2);
scatter(C(1,:),C(2,:),17,'filled')
xlabel('PC1'); ylabel('PC2');
```

Code Link: https://drive.google.com/drive/folders/16P4FzxDQJSkCY9kyo4qzb0DGWKfoicnr?usp=sharing